Git status interactive

Ben Klemens

12 December 2009

One of the first things that struck me as nice about Git was the status command, which produces something just shy of a script for revising the status of all the files. It even gives you tips about how to do common tasks.

I got even more excited when I saw `git rebase --interactive`, which generates a semi-script, opens it for you to edit, and then runs the thing automatically. That was smooth.

So I expected there'd be a similar procedure like `git status --interactive`, which, if it existed, would work like this:

- You type `git istatus`.
- Your favorite editor opens. There, you see the output from `git status`, plus instructions for some basic commands: put an `a` at the head of a line to add a file, an `i` to ignore it from now on, an `ea` to edit then add (which you'll do if you're merging), an `r` to remove the file from the repository, and so on.
- You exit, and your instructions are run.

Git doesn't do that. So I wrote a demo script to make that happen, git-status-interactive[1].

Click that link to save the script to your hard drive, and make it executable via the usual `chmod 755 git-status-interactive`. You probably want to alias the script using Git's aliasing system. For example, to allow the `git istatus` command I'd shown above, try this command from your bash prompt, in a single git repository:

```
git config --add alias.istatus \!/your/path/to/git-status-interactive
```

Or if you have the permissions to make global changes to the git config:

```
git config --global --add alias.istatus \!/your/path/to/git-status-interactive
```

Some further notes The script is a demo—dead simple, with no serious error checking. To some extent it's a feature request: Dear Git team, please implement something like this in Git, but competently. Also, dear readers, please drop me an email if you've improved this thing for the better.

[1]`https://github.com/b-k/git-status-interactive/blob/master/git-status-interactive`

[By the way, Git does have `git add -i`, which behaves very differently from the edit-a-generated-file mechanism from `git rebase --interactive`. `git add -i` doesn't let me tick off files to ignore, and doesn't help immensely during merging; though it will give you more control when adding, like committing changes to sections of a file.]

Apart from `git status` and the shell, I use exactly one program to make this happen: Sed. The prep step runs Sed to take in the output of `git status` and then remove non-comment lines and insert instructions; the post-editor step run Sed to replace the one-character markers with the full commands. That's all.

Because the modified file just runs as a shell script, you can add other commands as you prefer. For example, replacing the `\#` at the head of the line with an `rm` turns it into a standard remove command, or you can `mv` a file that git complains is in the wrong place (probably due to merging issues), et cetera.

In case you missed the link in the text above, download git-status-interactive[2] from Github.

[2] `https://github.com/b-k/git-status-interactive/blob/master/git-status-interactive`